OpenCV for Mechanical Engineering

Table of Contents

Introduction to OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and image processing library. It provides tools for real-time image analysis, object detection, and pattern recognition, making it highly useful in mechanical and manufacturing applications.

Applications of OpenCV in Mechanical Engineering

- Automated Quality Inspection
- Surface Defect Detection
- Object Tracking in Manufacturing
- Machine Vision for Robotics
- Measurement and Dimension Analysis

Running OpenCV in Google Colab

Google Colab provides a cloud-based environment to run OpenCV without requiring local installation. To upload images in Colab, use:

from google.colab import files

uploaded = files.upload()

2. Image Processing Basics

Loading and Displaying an Image

import cv2

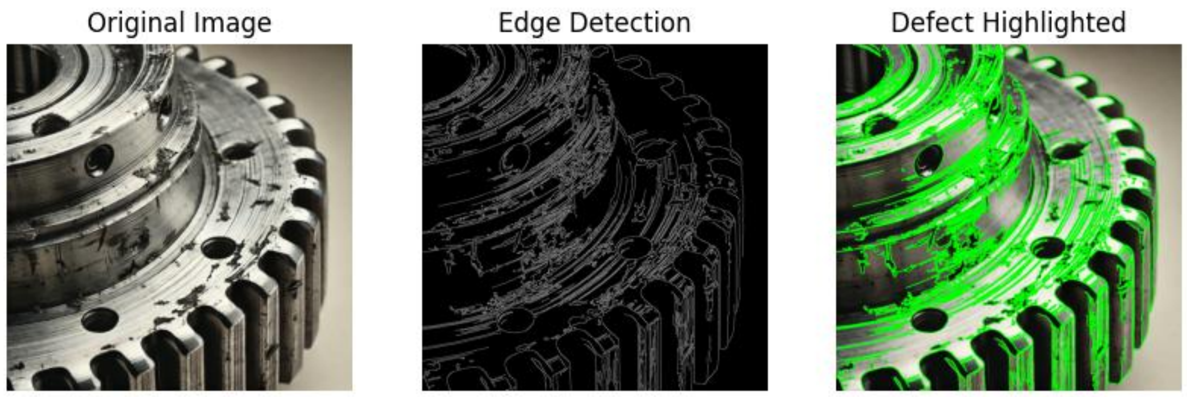
```
# Load an image from file
image = cv2.imread('sample_part.jpg')
# Display the image
cv2.imshow('Mechanical Component', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Mechanical Context: Used for visualizing machine components before analysis.

3. Converting to Grayscale and Applying Filters

Converting an Image to Grayscale

```
import cv2
# Load image in grayscale mode
image = cv2.imread('sample_part.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Grayscale Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

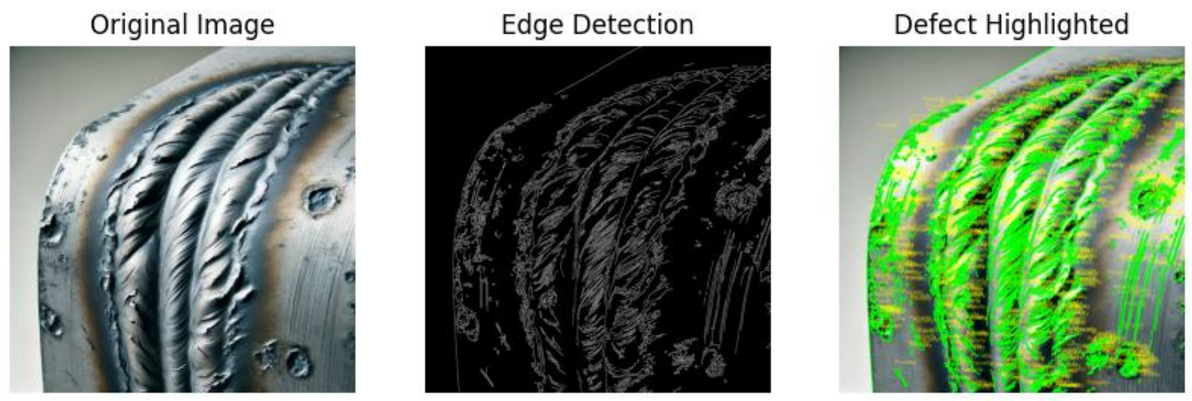


Mechanical Context: Helps in pre-processing images for defect detection.

Applying Gaussian Blur to Reduce Noise

```
import cv2
image = cv2.imread('sample_part.jpg')
blurred = cv2.GaussianBlur(image, (5,5), 0)
cv2.imshow('Blurred Image', blurred)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Mechanical Context: Used for reducing sensor noise in machine vision applications.



4. Edge Detection for Crack and Defect Analysis

Using Canny Edge Detection

```
import cv2

image = cv2.imread('sample_part.jpg', cv2.IMREAD_GRAYSCALE)

edges = cv2.Canny(image, 100, 200)

cv2.imshow('Edges', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Mechanical Context: Used for detecting cracks or defects on mechanical parts.

5. Object Detection for Quality Control

Detecting Contours in a Mechanical Part

```
import cv2

image = cv2.imread('sample_part.jpg')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

cv2.drawContours(image, contours, -1, (0, 255, 0), 2)

cv2.imshow('Contours', image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Mechanical Context: Helps in measuring dimensions and detecting defects in quality inspection.

6. Template Matching for Part Recognition

```
import cv2

import numpy as np

image = cv2.imread('factory_conveyor.jpg', 0)

template = cv2.imread('part_template.jpg', 0)

res = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)

thresh = 0.8

loc = np.where(res >= thresh)

for pt in zip(*loc[::-1]):

    cv2.rectangle(image, pt, (pt[0] + template.shape[1], pt[1] + template.shape[0]), (0, 255, 0), 2)

cv2.imshow('Detected Part', image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Mechanical Context: Used in automated assembly lines to recognize and verify parts.

7. Real-Time Object Tracking in Manufacturing

Tracking a Moving Object Using OpenCV

```
import cv2

cap = cv2.VideoCapture('conveyor_belt.mp4')

while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    edges = cv2.Canny(gray, 100, 200)

    cv2.imshow('Tracking Moving Parts', edges)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()
```

Mechanical Context: Used in manufacturing lines for tracking products and detecting anomalies.

8. Measurement and Dimension Analysis

Measuring an Object's Dimensions

```
import cv2

import numpy as np

image = cv2.imread('sample_part.jpg')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(gray, 100, 200)

contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

for cnt in contours:

    x, y, w, h = cv2.boundingRect(cnt)

    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('Measured Object', image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Mechanical Context: Used in automated systems for measuring manufactured parts.

Conclusion

OpenCV provides powerful tools for image processing and machine vision in mechanical engineering. From defect detection to automated quality control, OpenCV enhances the accuracy and efficiency of industrial processes. Understanding OpenCV applications in a mechanical context can lead to improved inspection, monitoring, and automation in manufacturing industries.

Additionally, OpenCV can be easily run in **Google Colab**, making it accessible for engineers to implement AI-driven solutions without requiring high-end hardware.

GitHub Repository

https://github.com/deepakrll/Open_CV_Programmes